Distributed Adaptive Node-Specific Signal Estimation in Fully Connected Sensor Networks—Part II: Simultaneous and Asynchronous Node Updating

Alexander Bertrand, *Student Member, IEEE*, and Marc Moonen, *Fellow, IEEE*

***Abstract*—In this paper, we revisit an earlier introduced distributed adaptive node-specific signal estimation (DANSE) algorithm that operates in fully connected sensor networks. In the original algorithm, the nodes update their parameters in a sequential round-robin fashion, which may yield a slow convergence of the estimators, especially so when the number of nodes in the network is large. When all nodes update simultaneously, the algorithm adapts more swiftly, but convergence can no longer be guaranteed. Simulations show that the algorithm then often gets locked in a suboptimal limit cycle. We first provide an extension to the DANSE algorithm, in which we apply an additional relaxation in the updating process. The new algorithm is then proven to converge to the optimal estimators when nodes update simultaneously or asynchronously, be it that the computational load at each node increases in comparison with the algorithm with sequential updates. Finally, based on simulations it is demonstrated that a simplified version of the new algorithm, without any extra computational load, can also provide convergence to the optimal estimators.**

***Index Terms*—Adaptive estimation, distributed estimation, wireless sensor networks (WSNs).**

I. Introduction

A wireless sensor network [1] consists of multiple sensor nodes that are connected with each other through a wireless link, and where each sensor node has its own processing unit. It allows to collect spatially diversed observations of a certain physical process, and to process these observations in a distributed fashion. A general objective is to utilize all sensor signal observations available in the entire network to perform a certain task, such as the estimation of a parameter or signal (see, for example, [2]–[10]).

In [9] and [10], we have introduced a distributed adaptive node-specific signal estimation (DANSE) algorithm that operates in a fully connected sensor network. The term "node-specific" refers to the fact that each node estimates a different desired signal. The nodes in the network broadcast compressed versions of their sensor signal observations, yet the algorithm converges to the optimal node-specific linear MMSE estimators, as if all sensor signal observations were available at each node, assuming that the desired signals of the different nodes share a common latent signal subspace with small dimension.

In the DANSE algorithm, as introduced in [9] and [10], nodes update in a sequential round-robin fashion. The algorithm typically converges in a small number of iterations (about two iterations per node). However, due to the sequential updating scheme, only one node at a time can estimate the statistics of its input signals and perform an update of its parameters. Since every such parameter update at a specific node changes the statistics of the node's broadcast signal, it takes some time before the next node can collect enough data to compute a reliable estimate of the modified signal statistics and then update its parameters. As a result, even though the DANSE algorithm converges in a small number of iterations, it may converge slowly in time, especially so when the number of nodes is large.

If alternatively, nodes would perform their updates simultaneously, the algorithm can adapt more swiftly, and all nodes can then estimate the signal statistics in parallel. However, convergence can no longer be guaranteed in this case, as will be shown by simulations. We will, therefore, extend the DANSE algorithm with a relaxation operation. We prove that this new algorithm converges when nodes update simultaneously or asynchronously. With the latter, we refer to the case where each node decides independently when and how often it updates its parameters, possibly simultaneously with other nodes. This avoids the need for a network-wide updating protocol that coordinates the updates between the different nodes.

We will also present a simplified version of the new algorithm, which reduces the computational load at each node. Although a theoretical convergence proof is not available for this simplified version of the algorithm, in simulations it is indeed observed to converge to the same optimal estimators. The tracking capabilities of the presented algorithm are tested in a simulated dynamic scenario, showing that simultaneous node updating significantly improves the adaptation speed of the DANSE algorithm, while maintaining its optimality.

Manuscript received October 21, 2009; accepted June 01, 2010. Date of publication June 10, 2010; date of current version September 15, 2010. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ta-Sung Lee. The work of A. Bertrand was supported by a Ph.D. grant of the I.W.T. (Flemish Institute for the Promotion of Innovation through Science and Technology). This work was carried out at the ESAT Laboratory of Katholieke Universiteit Leuven, in the frame of K.U. Leuven Research Council CoE EF/05/006 Optimization in Engineering (OPTEC), Concerted Research Action GOA-AMBioRICS, Concerted Research Action GOA-MaNet, the Belgian Programme on Interuniversity Attraction Poles initiated by the Belgian Federal Science Policy Office IUAP P6/04 (DYSCO, "Dynamical systems, control and optimization," 2007–2011), and Research Project FWO nr. G.0600.08 ("Signal processing and network design for wireless acoustic sensor networks"). The scientific responsibility is assumed by its authors.

The authors are with the Department of Electrical Engineering (ESAT-SCD/SISTA), Katholieke Universiteit Leuven, B-3001 Leuven, Belgium (e-mail: alexander.bertrand@esat.kuleuven.be; marc.moonen@esat.kuleuven.be).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2010.2052613

1053-587X/$26.00 © 2010 IEEE

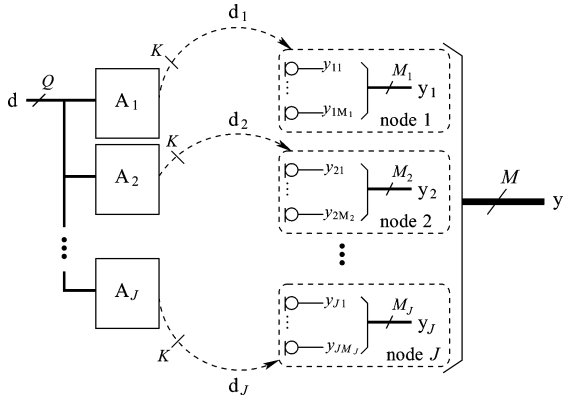

Fig. 1. Description of the scenario. The network contains J sensor nodes, $k = 1 \ldots J$, where node k collects M_k-channel sensor signal observations and estimates a node-specific desired signal $\mathbf{d}_k$, which is a mixture of the K channels of a common latent signal $\mathbf{d}$.

The paper is organized as follows. The problem formulation and notation are given in Section II. In Section III, we briefly review the DANSE algorithm of [9], [10]. In Section IV, we extend this algorithm with a relaxation operation to guarantee convergence and optimality when nodes update simultaneously or asynchronously, allowing for parallelization and uncoordinated computation. In Section V the convergence results are illustrated with numerical simulations in batch mode, and the tracking capabilities are demonstrated with an adaptive implementation of the algorithm. Conclusions are given in Section VI.

II. Problem Formulation and Notation

We consider an ideal fully connected network with sensor nodes $\{1, \ldots, J\} = \mathcal{J}$, in which data broadcast by a node can be captured by all other $(J-1)$ nodes in the network through an ideal link. Sensor node k collects observations of a complex[1] valued M_k-channel signal $\mathbf{y}_k[t]$, where $t \in \mathbb{N}$ is the discrete time index, and where $\mathbf{y}_k[t]$ is an M_k-dimensional column vector. Each channel $y_{kn}[t]$, $\forall n \in \{1, \ldots, M_k\}$, of the signal $\mathbf{y}_k[t]$ corresponds to a sensor signal to which node k has access. We assume that all signals are stationary[2] and ergodic. For the sake of an easy exposition, we will omit the time index when referring to a signal, and we will only write the time index when referring to one specific observation, i.e., $\mathbf{y}_k[t]$ is the observation of the signal $\mathbf{y}_k$ at time t. We define $\mathbf{y}$ as the M-channel signal in which all $\mathbf{y}_k$ are stacked, where $M = \sum_{k=1}^{J} M_k$. This scenario is depicted in Fig. 1.

We first consider the centralized estimation problem, i.e., we assume that each node has access to the observations of the entire M-channel signal $\mathbf{y}$. This corresponds to the case where nodes broadcast their uncompressed observations to all other nodes. In Sections III and IV, the general goal will be to compress the broadcast signals, while preserving the estimation performance of this centralized estimator. The objective for each node k is to optimally estimate a node-specific desired K-channel signal $\mathbf{d}_k$ that is correlated to $\mathbf{y}$. We consider the general case where $\mathbf{d}_k$ is not an observed signal, as it is the case in signal enhancement [11], [12]. As shown in Fig. 1, we assume that the node-specific desired signals $\mathbf{d}_k$ share a common Q-dimensional latent signal subspace, defined by an unknown Q-channel latent signal $\mathbf{d}$, i.e.

$$\mathbf{d}_k = \mathbf{A}_k \mathbf{d}, \quad \forall\, k \in \mathcal{J} \tag{1}$$

with $\mathbf{A}_k$ a full rank $K \times Q$ matrix with unknown coefficients. Without loss of generality, we assume that K is chosen equal to Q in the sequel. In many practical cases, only a subset of the channels of $\mathbf{d}_k$ may be of actual interest, in which case the other channels should be viewed as auxiliary channels to capture the entire Q-dimensional signal space spanned by all the desired signals of interest.

Node k uses a linear estimator $\mathbf{W}_k$ to estimate $\mathbf{d}_k$ as

$$\bar{\mathbf{d}}_k = \mathbf{W}_k^H \mathbf{y} \tag{2}$$

where $\mathbf{W}_k$ is a complex $M \times K$ matrix, and where superscript H denotes the conjugate transpose operator. We consider linear MMSE estimation based on a node-specific estimator $\hat{\mathbf{W}}_k$, i.e.

$$\hat{\mathbf{W}}_k = \arg\min_{\mathbf{W}_k} E\left\{\|\mathbf{d}_k - \mathbf{W}_k^H \mathbf{y}\|^2\right\} \tag{3}$$

where $E\{.\}$ denotes the expected value operator. The objective is to solve all J node-specific MMSE problems (3), i.e., one for each node. Assuming that the correlation matrix $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$ has full rank, the solution of (3) is

$$\hat{\mathbf{W}}_k = \mathbf{R}_{yy}^{-1} \mathbf{R}_{yd_k} \tag{4}$$

with $\mathbf{R}_{yd_k} = E\{\mathbf{y}\mathbf{d}_k^H\}$ [13]. Based on the assumed ergodicity, $\mathbf{R}_{yy}$ and $\mathbf{R}_{yd_k}$ can be estimated by time averaging. The $\mathbf{R}_{yy}$ is directly estimated from the sensor signal observations. A possible way to estimate $\mathbf{R}_{yd_k}$, where $\mathbf{d}_k$ is not an observed signal, is to use periodic training sequences, or by exploiting the ON-OFF behavior of the desired signal. The latter is often used in speech enhancement applications since speech signals contain pauses in between words and sentences [11], [12], [14]. In the sequel, we will assume that $\mathbf{R}_{yd_k}$ can be estimated during operation of the algorithm. Some example strategies to estimate $\mathbf{R}_{yd_k}$ can be found in [10].

Notice that each node k only has access to observations of $\mathbf{y}_k$ which is a subset of the channels of the full signal $\mathbf{y}$. Therefore, to find the optimal MMSE solution (4) in each node, the observations of $\mathbf{y}_k$ in principle have to be communicated to all nodes in the network, which requires a large communication bandwidth. However, the DANSE$_K$ algorithm, reviewed in the next section, significantly compresses this communication bandwidth, yet still achieves the optimal linear MMSE estimator (4) at each node.

III. The DANSE$_K$ Algorithm [9], [10]

In this section, we briefly review the DANSE$_K$ algorithm. For a more detailed description and analysis, we refer to [10]. In DANSE$_K$, the sensor nodes broadcast K linear combinations of their M_k-channel sensor signal observations. The DANSE$_K$

[1]Throughout this paper, all signals are assumed to be complex valued to permit frequency domain descriptions.

[2]In practice, the stationarity and ergodicity assumption can be relaxed to short-term stationarity and ergodicity, in which case the theory should be applied to finite signal segments that are assumed to be stationary and ergodic.

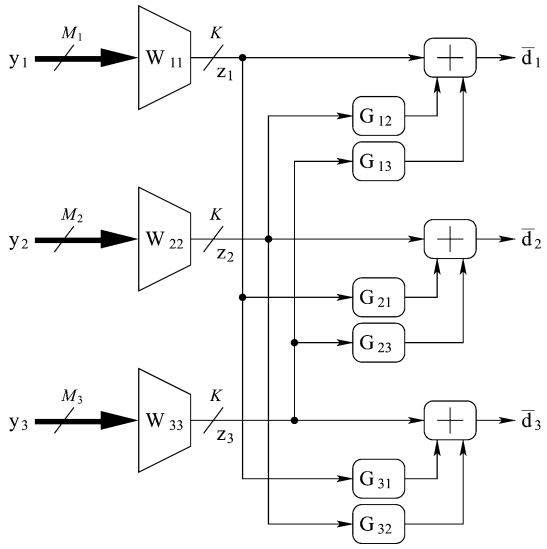

Fig. 2. The DANSE_K scheme with 3 nodes ($J = 3$). Each node k estimates a signal $\mathbf{d}_k$ using its own M_k-channel sensor signal observations, and two K-channel signals broadcast by the other two nodes.

algorithm thus yields a compression with a factor of M_k/K for the broadcasting by node k.

We define a partitioning of the estimator $\mathbf{W}_k$ as $\mathbf{W}_k = [\mathbf{W}_{k1}^T \cdots \mathbf{W}_{kJ}^T]^T$ with $\mathbf{W}_{kq}$ denoting the $M_k \times K$ submatrix of $\mathbf{W}_k$ that is applied to $\mathbf{y}_q$, and where superscript T denotes the transpose operator. In this way, (3) is equivalent to

$$\hat{\mathbf{W}}_k = \begin{bmatrix} \hat{\mathbf{W}}_{k1} \\ \vdots \\ \hat{\mathbf{W}}_{kJ} \end{bmatrix} \tag{5}$$

$$= \underset{\{\mathbf{W}_{k1},\ldots,\mathbf{W}_{kJ}\}}{\arg\min} \; E\left\{ \left\| \mathbf{d}_k - \sum_{q=1}^{J} \mathbf{W}_{kq}^H \mathbf{y}_q \right\|^2 \right\}. \tag{6}$$

In the DANSE_K algorithm, each node k broadcasts observations of the K-channel compressed signal $\mathbf{z}_k = \mathbf{W}_{kk}^H \mathbf{y}_k$ to the other nodes. Notice that $\mathbf{W}_{kk}$ thus both acts as a compressor and a part of the estimator $\mathbf{W}_k$. A node k can transform the K-channel signal $\mathbf{z}_q$ that it receives from another node q by a $K \times K$ transformation matrix $\mathbf{G}_{kq}$. Notice that, as $\mathbf{G}_{kq}$ is a square matrix, no decompression is involved. The parametrization of the $\mathbf{W}_k$ effectively applied at node k is therefore

$$\tilde{\mathbf{W}}_k = \begin{bmatrix} \mathbf{W}_{11}\mathbf{G}_{k1} \\ \vdots \\ \mathbf{W}_{JJ}\mathbf{G}_{kJ} \end{bmatrix}. \tag{7}$$

We use a tilde to indicate that the estimator is parametrized according to (7). In this parametrization, node k can only control the parameters $\mathbf{W}_{kk}$ and $\mathbf{G}_k = [\mathbf{G}_{k1}^T \cdots \mathbf{G}_{kJ}^T]^T$. To remove the ambiguity in $\mathbf{W}_{kk}\mathbf{G}_{kk}$, we assume that $\mathbf{G}_{kk} = \mathbf{I}_K$ with $\mathbf{I}_K$ denoting the $K \times K$ identity matrix. A schematic illustration of this scheme in a 3-node network ($J = 3$), is shown in Fig. 2. The goal of the DANSE_K algorithm is to iteratively update the parameters of (7) until $\left(\tilde{\mathbf{W}}_1, \ldots, \tilde{\mathbf{W}}_J\right) = \left(\hat{\mathbf{W}}_1, \ldots, \hat{\mathbf{W}}_J\right)$.

In the sequel, we will use the following notation and definitions. In general, we will use X^i to denote X at iteration i, where X can be a signal or a parameter. We let $\mathbf{z} = \left[\mathbf{z}_1^T \cdots \mathbf{z}_J^T\right]^T$, and we use the notation $\mathbf{z}_{-k} = \left[\mathbf{z}_1^T \ldots \mathbf{z}_{k-1}^T \mathbf{z}_{k+1}^T \ldots \mathbf{z}_J^T\right]^T$, i.e., the vector $\mathbf{z}$ without the subvector $\mathbf{z}_k$. Similarly, we will use $\mathbf{G}_{k-q}$ to denote the matrix $\mathbf{G}_k$ without $\mathbf{G}_{kq}$.

The DANSE_K algorithm will iteratively update the parameters in (7) by solving local MMSE problems at each node $k \in \mathcal{J}$. For node k, at iteration i, this update is computed as

$$\begin{bmatrix} \mathbf{W}_{kk}^{i+1} \\ \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \underset{\mathbf{W}_{kk},\mathbf{G}_{k-k}}{\arg\min} \; E\left\{ \left\| \mathbf{d}_k - [\,\mathbf{W}_{kk}^H \mathbf{G}_{k-k}^H\,] \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k}^i \end{bmatrix} \right\|^2 \right\}. \tag{8}$$

In this local MMSE problem, the node-specific desired signal $\mathbf{d}_k$ is estimated by means of the input signals of node k, i.e., its M_k-channel sensor signal $\mathbf{y}_k$ and the $K(J-1)$-channel signal $\mathbf{z}_{-k}$ containing the broadcast signals of the other nodes. Let $\tilde{\mathbf{y}}_k^i$ denote the stacked version of these local input signals at node k, i.e.

$$\tilde{\mathbf{y}}_k^i = \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k}^i \end{bmatrix}. \tag{9}$$

Then the solution of (8) is

$$\begin{bmatrix} \mathbf{W}_{kk}^{i+1} \\ \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \left(\mathbf{R}_{\tilde{y}_k \tilde{y}_k}^i\right)^{-1} \mathbf{R}_{\tilde{y}_k d_k}^i \tag{10}$$

with

$$\mathbf{R}_{\tilde{y}_k \tilde{y}_k}^i = E\{\tilde{\mathbf{y}}_k^i \tilde{\mathbf{y}}_k^{i\,H}\} \tag{11}$$

$$\mathbf{R}_{\tilde{y}_k d_k}^i = E\{\tilde{\mathbf{y}}_k^i \mathbf{d}_k^H\}. \tag{12}$$

We define a block size B which denotes the number of observations that the nodes collect in between two successive node updates, i.e., in between two increments of i. The DANSE_K algorithm now consists of the following steps:

1) Initialize: $i \leftarrow 0$, $u \leftarrow 1$
   Initialize $\mathbf{W}_{kk}^0$ and $\mathbf{G}_{k-k}^0$ with random matrices, $\forall\, k \in \mathcal{J}$.
2) Each node $k \in \mathcal{J}$ performs the following operation cycle:
   - Collect the sensor observations $\mathbf{y}_k[iB + n]$, $n = 0 \ldots B-1$.
   - Compress these M_k-dimensional observations to K-dimensional vectors

$$\mathbf{z}_k^i[iB+n] = \mathbf{W}_{kk}^{i\,H} \mathbf{y}_k[iB+n], \quad n = 0 \ldots B-1. \tag{13}$$

   - Broadcast the compressed observations $\mathbf{z}_k^i[iB+n]$, $n = 0 \ldots B-1$, to the other nodes.
   - Collect the $K(J-1)$-dimensional data vectors $\mathbf{z}_{-k}^i[iB+n]$, $n = 0 \ldots B-1$, which are stacked versions of the compressed observations received from the other nodes.
   - Update the estimates of $\mathbf{R}_{\tilde{y}_k \tilde{y}_k}^i$ and $\mathbf{R}_{\tilde{y}_k d_k}^i$, by including the newly collected data.
   - Update the node-specific parameters

$$\begin{bmatrix} \mathbf{W}_{kk}^{i+1} \\ \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \begin{cases} \left(\mathbf{R}_{\tilde{y}_k \tilde{y}_k}^i\right)^{-1} \mathbf{R}_{\tilde{y}_k d_k}^i & \text{if } k = u \\ \begin{bmatrix} \mathbf{W}_{kk}^i \\ \mathbf{G}_{k-k}^i \end{bmatrix} & \text{if } k \neq u. \end{cases} \tag{14}$$

   - Compute the estimate of $\mathbf{d}_k[iB+n]$ as

$$\bar{\mathbf{d}}_k[iB+n] = \mathbf{W}_{kk}^{i+1\,H} \mathbf{y}_k[iB+n] + \mathbf{G}_{k-k}^{i+1\,H} \mathbf{z}_{-k}^i[iB+n]. \tag{15}$$

3) $i \leftarrow i+1$
4) $u \leftarrow (u \bmod J)+1$
5) return to step 2

Remark I: Notice that the different iterations are spread out over time. Therefore, the iterative characteristics of the algorithm do not have an impact on the amount of data that is transmitted, i.e., each sample is only broadcast once since the time index in (13) and (15) shifts together with the iteration index. Therefore, an update of the estimator parameters only has an impact on future samples and old samples are not re-estimated.

Remark II: For implementation aspects regarding the estimation of the correlation matrices $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}$ and $\mathbf{R}^i_{\tilde{y}_k d_k}$, we refer to [10].

It is noted that the nodes update in a sequential round robin fashion. At each iteration, one specific node estimates the signal statistics of its input channels, and updates its parameters by optimizing its local node-specific estimation problem (8), while the parameters at the other nodes are freezed. The following theorem guarantees convergence of the DANSE$_K$ algorithm to the optimal estimators[3]:

Theorem III.1: If the sensor signal correlation matrix $\mathbf{R}_{yy}$ has full rank, and if (1) is satisfied with $K = Q$, then for the DANSE$_K$ algorithm as described above, the sequence $\left(\tilde{\mathbf{W}}^i_k\right)_{i\in\mathbb{N}}$ converges to the optimal solution (4), $\forall\, k \in \mathcal{J}$, for any initialization of the parameters.

Proof: See [10]. ∎

IV. Simultaneous and Uncoordinated Updating

As mentioned in Section III, the nodes in the DANSE$_K$ algorithm update their parameters in a sequential round-robin fashion. When node k performs an update at iteration i, the signal $\mathbf{z}^i_k$ changes to $\mathbf{z}^{i+1}_k$, which generally has different statistics. Therefore, the next node to perform an update needs sufficient time to collect a sufficient number of observations of $\mathbf{z}^{i+1}_k$ to reliably estimate the correlation coefficients involving this signal. Therefore, even though the DANSE$_K$ algorithm converges fast in terms of number of iterations, it may converge rather slowly in time, which affects its tracking performance.

This problem obviously becomes worse when the number of nodes is large, such that one round of updates takes a long time. We may expect that the network can react much faster to changes in the environment when nodes would be able to update simultaneously. Simulations in Section V will show that convergence is indeed faster in this case, both in time and in number of iterations. Unfortunately, these simulations will also show that convergence is no longer guaranteed. Therefore, in this section, we first extend the DANSE$_K$ algorithm, to restore convergence, and then consider a number of algorithmic simplifications.

A. The S-DANSE$_K$ Algorithm

Consider the case where all nodes update simultaneously, i.e., (10) is applied for all k simultaneously in iteration i. We refer to this as simultaneous-DANSE$_K$ or S-DANSE$_K$. In a network with two nodes ($J = 2$), convergence of DANSE$_K$ under sequential updating also implies convergence[4] of S-DANSE$_K$. Unfortunately, this is not always true if $J > 2$. Extensive simulations show that the S-DANSE$_K$ algorithm does not always converge to a stable estimator, but may get locked in a suboptimal limit cycle (see Fig. 5). This means that the parameters at the different nodes keep switching between multiple suboptimal estimators. The occurrence of these limit cycles heavily depends on the scenario, but a clear rule to predict whether the S-DANSE$_K$ algorithm will converge to a stable estimator or get locked in a limit cycle, has not been found. In the white noise experiments described in Section V-A, the S-DANSE$_K$ algorithm mostly converges to the optimal solution. However, in the scenario of Section V-B, and in the simulations in acoustic sensor networks described in [12], limit cycle behavior has been observed quite frequently.

The reason why S-DANSE$_K$ often fails to converge can be intuitively explained as follows. Assume that node k computes (10), i.e., it optimizes its estimators with respect to the current statistics of the broadcast signals in $\mathbf{z}^i_{-k}$. If the other nodes update simultaneously, all the signals in $\mathbf{z}^i_{-k}$ immediately switch to $\mathbf{z}^{i+1}_{-k}$. Since the newly applied estimator $\tilde{\mathbf{W}}^{i+1}_k$ was optimized with respect to the signal statistics of $\mathbf{z}^i_{-k}$, it immediately becomes suboptimal again due to simultaneous updates of the other nodes. In fact, the MSE of the node-specific estimators may therefore increase after the update, i.e., the new estimator $\bar{\mathbf{d}}^{i+1}_k = \tilde{\mathbf{W}}^{i+1\,H}_k \mathbf{y}$ for node k may be worse than the old estimator $\bar{\mathbf{d}}^i_k = \tilde{\mathbf{W}}^{i\,H}_k \mathbf{y}$ before the update.

This fundamental difference between the convergence properties of DANSE$_K$ and S-DANSE$_K$ is similar to the difference between the convergence properties of nonlinear Gauss-Seidel iteration and nonlinear Jacobi iteration, as described in [16]. In both the Gauss-Seidel and Jacobi procedures, subsequent optimization steps are performed over a subset of the variables of an objective function, while keeping the other variables fixed. However, both methods differ in the way they update the new iteration point. In Gauss-Seidel iteration, the iteration point is immediately updated after the optimization of a single subset of variables, whereas in Jacobi iteration, the actual iteration point is updated after an optimization of all the subsets simultaneously with respect to the current iteration point. In particular, for a cost function $f(\mathbf{w})$ with $\mathbf{w} = (w_1, w_2)$, the nonlinear Gauss-Seidel and Jacobi procedure are as follows (both are illustrated in Fig. 3):

Nonlinear Gauss-Seidel:

1) Initialize $\mathbf{w}^0 = (w_1^0, w_2^0)$ randomly.
2) $i \leftarrow 0$
3) $\begin{cases} w_1^{i+1} \leftarrow \arg\min_{w_1} f(w_1, w_2^i) \\ w_2^{i+1} \leftarrow w_2^i \\ w_1^{i+2} \leftarrow w_1^{i+1} \\ w_2^{i+2} \leftarrow \arg\min_{w_2} f(w_1^{i+1}, w_2) \end{cases}$
4) $i \leftarrow i+2$
5) Return to step 3.

[3]Theorem III.1, and all convergence theorems in the sequel, assume that the signal statistics are perfectly known by the algorithm, i.e., as if an infinite observation window is used. In other words, estimation errors in $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}$ and $\mathbf{R}^i_{\tilde{y}_k d_k}$ are not taken into account.

[4]This can be shown by similar arguments as in [15], where it is explained that convergence of Gauss-Seidel iteration implies convergence of Jacobi iteration in the 2-node case.

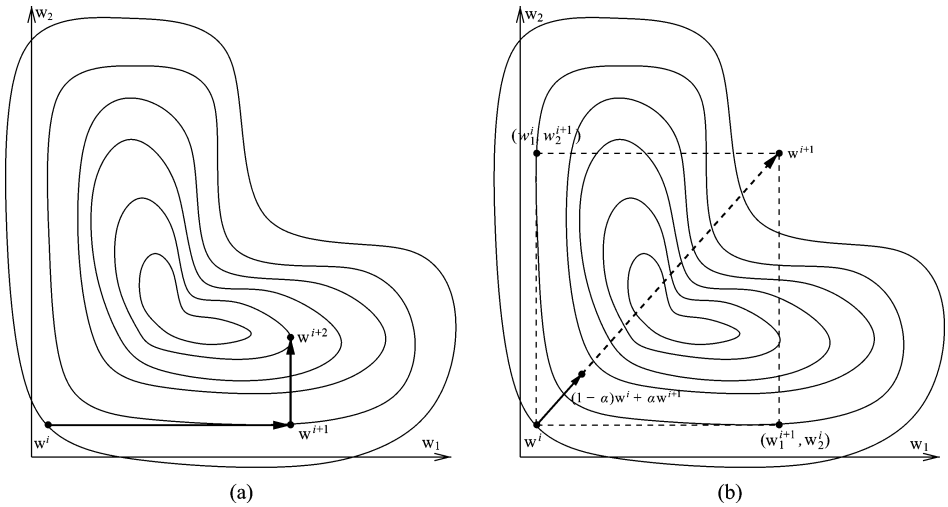

Fig. 3. Illustration of nonlinear Gauss-Seidel iteration and nonlinear Jacobi iteration. (a) Nonlinear Gauss-Seidel iteration. (b) Nonlinear Jacobi iteration and its relaxed version.

Nonlinear Jacobi:

1) Initialize $\mathbf{w}^0 = (w_1^0, w_2^0)$ randomly.
2) $i \leftarrow 0$.
3) $\begin{cases} w_1^{i+1} \leftarrow \arg\min_{w_1} f(w_1, w_2^i) \\ w_2^{i+1} \leftarrow \arg\min_{w_2} f(w_1^i, w_2) \end{cases}$
4) $i \leftarrow i + 1$
5) Return to step 3.

It is obvious that the Gauss-Seidel procedure will always result in a decrease of the objective function in each iteration, i.e., $f(\mathbf{w}^{i+1}) \leq f(\mathbf{w}^i)$, and therefore convergence is generally not an issue. In the Jacobi procedure, however, this argument fails since an update can result in $f(\mathbf{w}^{i+1}) > f(\mathbf{w}^i)$, as illustrated in Fig. 3(b). This is because each variable is optimized with respect to the previous value of the other variables, which are not retained after the update. Therefore, Jacobi iteration often fails to converge[5].

The DANSE_K algorithm sequentially optimizes a specific element of $\{\mathbf{W}_{11}, \ldots, \mathbf{W}_{JJ}\}$ with respect to the current values of the other elements (corresponding to the other nodes). Notice that this is akin to Gauss-Seidel iteration. From the same point of view, the S-DANSE_K algorithm is akin to Jacobi iteration since the elements in $\{\mathbf{W}_{11}, \ldots, \mathbf{W}_{JJ}\}$ are again optimized with respect to the current values of the other elements, but all of them are updated simultaneously. This relationship illustrates why S-DANSE_K often fails to converge whereas DANSE_K always converges to the optimal estimators. However, keep in mind that the similarity between DANSE_K and S-DANSE_K on the one hand and the Gauss-Seidel and Jacobi procedure on the other hand, only holds up to a certain level. Indeed, in DANSE_K and S-DANSE_K none of the variables are actually fixed. If node k optimizes $\mathbf{W}_{kk}$, it can also partially manipulate the $\mathbf{W}_{qq}$ of other nodes $q \neq k$ by means of the variable $\mathbf{G}_{kq}$. Furthermore, the objective function is different at each node, and therefore each element in $\{\mathbf{W}_{11}, \ldots, \mathbf{W}_{JJ}\}$ is optimized with respect to a different cost function.

[5]In the illustration of Fig. 3, the Jacobi iteration eventually will converge since the variables of the cost function are divided into two subsets ($\{w_1\}$ and $\{w_2\}$). Jacobi iteration always converges in this case [15]. However, this is generally not true when more than 2 variable subsets are used [16].

B. The rS-DANSE_K^+ Algorithm

In the previous subsection, we explained that the S-DANSE_K algorithm often fails to converge since each node optimizes its estimators with respect to signal statistics that immediately become invalid after the update. The idea is now to partially counter these dynamics by letting each node k perform a relaxed update of its partial estimator $\mathbf{W}_{kk}$ to a convex combination of $\mathbf{W}_{kk}^i$ and $\mathbf{W}_{kk}^{i+1}$, i.e., $(1-\alpha)\mathbf{W}_{kk}^i + \alpha\mathbf{W}_{kk}^{i+1}$ with $\alpha \in (0, 1]$. This allows the old estimator to remain partially active and to generate broadcast signals $\mathbf{z}_{-k}^{i+1}$ that have partial components equal to the old broadcast signals $\mathbf{z}_{-k}^i$. This corresponds to some first order memory in the updating dynamics[6], which is absent in the S-DANSE_K algorithm. The intuition behind this relaxation procedure is illustrated in Fig. 3(b), where it is observed that relaxation dampens the Jacobi-iteration. If α is chosen small enough, the objective function decreases due to the relaxed update.

In this section, we will modify the S-DANSE_K algorithm accordingly, to enforce convergence when nodes update simultaneously. Although this procedure may appear to be quite heuristic at first sight, it can be theoretically justified, i.e., we will prove that the new algorithm enforces convergence if the relaxation stepsize satisfies certain properties.

Before describing the algorithm, we introduce some additional notation. The matrix $\mathbf{W}$ (without subscript) denotes the stacked matrix of all $\mathbf{W}_{kk}$ matrices, i.e.

$$\mathbf{W} = [\,\mathbf{W}_{11}^T \quad \mathbf{W}_{22}^T \quad \ldots \quad \mathbf{W}_{JJ}^T\,]^T. \tag{16}$$

We also define the MSE cost functions corresponding to node k, namely

$$J_k(\mathbf{W}_k) = E\left\{\|\mathbf{d}_k - \mathbf{W}_k^H\mathbf{y}\|^2\right\} \tag{17}$$

[6]The same technique is sometimes used in game theory to enforce convergence to a Nash equilibrium when all players simultaneously perform a best-reply to the current strategy setting (see e.g., [15], [17]). However, this technique only works for specific games that satisfy certain assumptions, which are mostly very technical and stringent.

as used in (3), and

$$\tilde{J}_k(\mathbf{W}, \mathbf{G}_k) = J_k(\tilde{\mathbf{W}}_k) \tag{18}$$

where $\tilde{\mathbf{W}}_k$ is defined from $\mathbf{W}$ and $\mathbf{G}_k$ as in (7). Notice that $\mathbf{G}_k$ contains the entry $\mathbf{G}_{kk}$, which is never actually computed in the original DANSE_K algorithm. We define $F_k(\mathbf{W}^i)$ as the function that generates $\mathbf{W}_{kk}^{i+1}$ according to (10), i.e.

$$F_k(\mathbf{W}^i) = \mathbf{W}_{kk}^{i+1} \tag{19}$$

$$= [\,\mathbf{I}_{M_k} \quad \mathbf{O}_{M_k \times K(J-1)}\,] \left(\mathbf{R}_{\tilde{y}_k \tilde{y}_k}^i\right)^{-1} \mathbf{R}_{\tilde{y}_k d_k}^i \tag{20}$$

with $\mathbf{O}_{P\times Q}$ denoting an all-zero $P \times Q$ matrix. It is noted that the right-hand side (RHS) of (20) depends on all entries of the argument $\mathbf{W}^i$ through the signal $\mathbf{z}_{-k}^i$, which is not explicitly revealed in this expression. We also define the correlation matrices: $\mathbf{R}_{zz}^i = E\{\mathbf{z}^i \mathbf{z}^{i\,H}\}$ and $\mathbf{R}_{zd_k}^i = E\{\mathbf{z}^i \mathbf{d}_k^H\}$.

Now consider the following algorithm:

1) Initialize: $i \leftarrow 0$
   Initialize $\mathbf{W}_{kk}^0$ and $\mathbf{G}_{k-k}^0$ with random matrices, $\forall\, k \in \mathcal{J}$.
2) Each node $k \in \mathcal{J}$ performs the following operation cycle simultaneously:
   - Collect the sensor observations $\mathbf{y}_k[iB + n]$, $n = 0 \ldots B-1$.
   - Compress these M_k-dimensional observations to K-dimensional vectors

$$\mathbf{z}_k^i[iB+n] = \mathbf{W}_{kk}^{i\,H} \mathbf{y}_k[iB+n], \quad n = 0 \ldots B-1. \tag{21}$$

   - Broadcast the compressed observations $\mathbf{z}_k^i[iB+n]$, $n = 0 \ldots B-1$, to the other nodes.
   - Collect the $K(J-1)$-dimensional data vectors $\mathbf{z}_{-k}^i[iB+n]$, $n = 0 \ldots B-1$, which are stacked versions of the compressed observations received from the other nodes.
   - Update the estimates of $\mathbf{R}_{\tilde{y}_k \tilde{y}_k}^i$, $\mathbf{R}_{zz}^i$, $\mathbf{R}_{\tilde{y}_k d_k}^i$, and $\mathbf{R}_{zd_k}^i$ by including the newly collected data.
   - Compute a new estimate for $\mathbf{G}_k^i$

$$\mathbf{G}_k^{i+1} = \arg\min_{\mathbf{G}_k} \tilde{J}_k\left(\mathbf{W}^i, \mathbf{G}_k\right) \tag{22}$$

$$= \left(\mathbf{R}_{zz}^i\right)^{-1} \mathbf{R}_{zd_k}^i. \tag{23}$$

   - Choose an $\alpha^i \in (0, 1]$, and compute a new estimate for $\mathbf{W}_{kk}^i$

$$\mathbf{W}_{kk}^{i+1} = (1-\alpha^i)\mathbf{W}_{kk}^i \mathbf{G}_{kk}^{i+1} + \alpha^i F_k(\mathbf{W}^i). \tag{24}$$

   - Compute the estimate of $\mathbf{d}_k[iB+n]$ as

$$\bar{\mathbf{d}}_k[iB+n] = \mathbf{W}_{kk}^{i+1\,H} \mathbf{y}_k[iB+n] + \mathbf{G}_{k-k}^{i+1\,H} \mathbf{z}_{-k}^i[iB+n]. \tag{25}$$

3) $i \leftarrow i+1$
4) return to step 2

We will refer to this algorithm as relaxed simultaneous-DANSE_K^+ or rS-DANSE_K^+, where the superscript $+$ is added because of the extra optimization (22). Notice that in rS-DANSE_K^+, the variable $\mathbf{G}_{kk}^{i+1}$ is indeed explicitly computed, unlike in the DANSE_K algorithm with sequential updating. However, it is merely applied as a transformation for $\mathbf{W}_{kk}^i$ before the relaxed update (24). After the update (24), the $\mathbf{G}_{kk}$ that is actually applied to $\mathbf{W}_{kk}$ in the parametrization (7) is again fixed to an identity matrix, i.e., the schematic representation shown in Fig. 2, in which the $\mathbf{G}_{kk}$'s are omitted, also holds for the rS-DANSE_K^+ algorithm.

Due to the strict convexity of the cost function $J_k(\mathbf{W}_k)$, the simultaneous update function $F(\mathbf{W}) = \left[F_1(\mathbf{W})^T \ldots F_J(\mathbf{W})^T\right]^T$ has only one fixed point satisfying $\mathbf{W} = F(\mathbf{W})$, namely the optimal solution $\hat{\mathbf{W}}$ to which DANSE_K converges if sequential updating is used. Notice that for any α^i, this point $\hat{\mathbf{W}}$ is also the equilibrium point of (22)–(24). The choice of α^i is critical and decides whether or not (22)–(24) converges to this fixed point. The following theorem presents a strategy for choosing the stepsize parameter α^i, that guarantees convergence to the optimal solution:

Theorem IV.1: If the sensor signal correlation matrix $\mathbf{R}_{yy}$ has full rank, and if (1) is satisfied with $K = Q$, then for the rS-DANSE_K^+ algorithm as aforementioned, with stepsizes α^i satisfying

$$\alpha^i \in (0, 1] \tag{26}$$

$$\lim_{i \to \infty} \alpha^i = 0 \tag{27}$$

$$\sum_{i=0}^{\infty} \alpha^i = \infty \tag{28}$$

the sequence $\left(\tilde{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ converges to the optimal solution (4), $\forall\, k \in \mathcal{J}$, for any initialization of the parameters.

Proof: See Appendix A. ∎

A possible choice for the sequence $\left(\alpha^i\right)_{i\in\mathbb{N}}$ is $\alpha^i = 1/i$ or a slower decreasing sequence, e.g., $\alpha^i = 1/\log_{10}(10+i)$. The conditions on the sequence $\left(\alpha^i\right)_{i\in\mathbb{N}}$ in Theorem IV.1 are however quite conservative. Extensive simulations indicate that there appears to exist a critical $\hat{\alpha}$ and a corresponding i^* such that rS-DANSE_K^+ converges as long as $\alpha^i < \hat{\alpha}, \forall\, i > i^*$. Since this critical $\hat{\alpha}$ is generally not known *a priori*, a good strategy consists in initially choosing $\alpha^i = 1$, i.e., no relaxation, in combination with a limit cycle detector. Only when a limit cycle is detected, α^i is decreased until the limit cycle disappears. The algorithm then converges with fixed α^i to an equilibrium point of F, i.e., the matrix $\mathbf{W}$ that corresponds to solution (4). It is noted that, even if the above mentioned critical $\hat{\alpha}$ does not exist, this procedure automatically guarantees that the conditions (26)–(28) on the stepsize α^i are satisfied, and therefore the algorithm will converge under all circumstances.

C. The rS-DANSE_K Algorithm

Update rule (22) in rS-DANSE_K^+ increases the computational load in every node, since it represents an extra MSE minimization in addition to the implicit MSE minimization in $F_k(\mathbf{W}^i)$. However, extensive simulations indicate that this extra MSE minimization is not crucial to enforce convergence. The $\mathbf{G}_{k-k}^{i+1}$ in (10) that are generated as a by-product in the evaluation of

$F_k(\mathbf{W}^i)$ in (20), may be used instead. We will refer to this modification as rS-DANSE$_K$, without the superscript $+$.

The rS-DANSE$_K$ algorithm consists of the following steps:

1) Initialize: $i \leftarrow 0$
   Initialize $\mathbf{W}_{kk}^0$ and $\mathbf{G}_{k-k}^0$ with random matrices, $\forall\, k \in \mathcal{J}$.
2) Each node $k \in \mathcal{J}$ performs the following operation cycle simultaneously:
   - Collect the sensor observations $\mathbf{y}_k[iB + n]$, $n = 0 \ldots B-1$.
   - Compress these M_k-dimensional observations to K-dimensional vectors
   $$\mathbf{z}_k^i[iB+n] = \mathbf{W}_{kk}^{i\,H}\mathbf{y}_k[iB+n], \quad n = 0 \ldots B-1. \tag{29}$$
   - Broadcast the compressed observations $\mathbf{z}_k^i[iB+n], n = 0 \ldots B-1$, to the other nodes.
   - Collect the $K(J-1)$-dimensional data vectors $\mathbf{z}_{-k}^i[iB+n]$, $n = 0 \ldots B-1$, which are stacked versions of the compressed observations received from the other nodes.
   - Update the estimates of $\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i$ and $\mathbf{R}_{\tilde{y}_k d_k}^i$ by including the newly collected data.
   - Compute
   $$\begin{bmatrix} \mathbf{W}_{kk}^{\text{temp}} \\ \mathbf{G}_{k-k}^{i+1} \end{bmatrix} = \left(\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i\right)^{-1}\mathbf{R}_{\tilde{y}_k d_k}^i. \tag{30}$$
   - Choose an $\alpha^i \in (0,1]$, and compute a new estimate for $\mathbf{W}_{kk}^i$:
   $$\mathbf{W}_{kk}^{i+1} = (1-\alpha^i)\mathbf{W}_{kk}^i + \alpha^i \mathbf{W}_{kk}^{\text{temp}}. \tag{31}$$
   - Compute the estimate of $\mathbf{d}_k[iB+n]$ as
   $$\bar{\mathbf{d}}_k[iB+n] = \mathbf{W}_{kk}^{i+1\,H}\mathbf{y}_k[iB+n] + \mathbf{G}_{k-k}^{i+1\,H}\mathbf{z}_{-k}^i[iB+n]. \tag{32}$$
3) $i \leftarrow i+1$
4) return to step 2

Even in cases where S-DANSE$_K$ results in a suboptimal limit cycle, the rS-DANSE$_K$ algorithm is observed to converge to the optimal solution (4) under similar conditions as the rS-DANSE$_K^+$ algorithm, i.e., if α^i satisfies (26)–(28), or if it becomes smaller than a critical value. This is stated here as an observation based on extensive simulation (see Section V, and [12]), but without a formal proof[7]. Notice that the extra optimization (22) generally speeds up the convergence, especially so when α^i is small, which makes rS-DANSE$_K^+$ faster than rS-DANSE$_K$ (see Section V). However, this faster convergence is mostly insignificant compared to the increase in convergence speed that is obtained by letting nodes update simultaneously instead of sequentially. Therefore, it may be desirable to use rS-DANSE$_K$ instead of rS-DANSE$_K^+$, to decrease the computational load at each node.

D. Asynchronous Updating

The DANSE$_K$ algorithm and its variations described in the previous sections, all imply the need for a network wide update protocol that coordinates the updating of nodes in the network. Here we consider the case in which the updating happens in an asynchronous[8] fashion, i.e., nodes can decide independently when and how often they update their parameters and nodes do not know when the other nodes perform an update. This can be viewed as the hybrid case between simultaneous updating and sequential round-robin updating.

The iterations in which node k updates its parameters is given by a binary sequence $\left(s_k^i\right)_{i\in\mathbb{N}}$, with $s_k^i \in \{0,1\}$, where $s_k^i = 1$ implies that node k performs an update at iteration i and where $s_k^i = 0$ implies that node k does not perform an update at iteration i. We will assume that

$$\forall\, k \in \mathcal{J} : \sum_{i=0}^{\infty} s_k^i = \infty \tag{33}$$

which means that none of the nodes permanently stops the updating process of its parameters. It is noted that the sequences $\left(s_k^i\right)_{i\in\mathbb{N}}$, $\forall\ k \in \mathcal{J}$, can be totally random and they are not known to the other nodes $q \neq k$. Therefore, the updates of each node can happen *ex tempore*, i.e., each node can decide on the fly when and how often it performs an update of its parameters. By setting the block size to $B = 1$, the iteration index i coincides with the time index t. This models the case where nodes are allowed to perform an update at any[9] sampling time t, i.e., fully asynchronously.

The relaxed asynchronous-DANSE$_K^+$ or rA-DANSE$_K^+$ algorithm is defined equivalently to rS-DANSE$_K^+$, but the update (23)–(24) is replaced with

$$\mathbf{G}_k^{i+1} = \begin{cases} \left(\mathbf{R}_{zz}^i\right)^{-1}\mathbf{R}_{zd_k}^i & \text{if } s_k^i = 1 \\ \mathbf{G}_k^i & \text{if } s_k^i = 0 \end{cases} \tag{34}$$

$$\mathbf{W}_{kk}^{i+1} = \begin{cases} (1-\alpha^i)\mathbf{W}_{kk}^i\mathbf{G}_{kk}^{i+1} + \alpha^i F_k(\mathbf{W}^i) & \text{if } s_k^i = 1 \\ \mathbf{W}_{kk}^i & \text{if } s_k^i = 0. \end{cases} \tag{35}$$

Theorem IV.2: If the sensor signal correlation matrix $\mathbf{R}_{yy}$ has full rank, and if (1) is satisfied with $K = Q$, then for the rA-DANSE$_K^+$ algorithm as described above, with stepsizes α^i satisfying (26)–(28) and sequences $\left(s_k^i\right)_{i\in\mathbb{N}}$ satisfying (33), the sequence $\left(\tilde{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ converges to the optimal solution (4), $\forall\, k \in \mathcal{J}$, for any initialization of the parameters.

Proof: The proof is a straightforward modification of the proof of Theorem IV.1, as given in Appendix A. ■

The rA-DANSE$_K^+$ algorithm can also be simplified to the rA-DANSE$_K$ algorithm, similarly to the rS-DANSE$_K$ algorithm as described in Section IV-C.

V. Numerical Simulations

A. Batch Mode Simulations

In this section, we simulate the different algorithms mentioned in this paper in batch mode. This means that all itera-

[7]Due to a subtle difference, some parts of the proof of Theorem IV.1 are not applicable to the case of rS-DANSE$_K$. However, the main idea behind relaxation, as illustrated in Fig. 3(b), remains.

[8]The term 'asynchronous' here refers to the fact that there is no synchronization at the iteration level. However, at the sample level, an accurate synchronization of the sample clocks at the different nodes is still required.

[9]It is noted that setting $B = 1$ does not imply that nodes update at *every* sampling time t. As aforementioned, it is usually better to leave some time between successive updates of a certain node, i.e., using sparse sequences $(s_k^i)_{i\in\mathbb{N}}$.

tions are performed on the full signal length, including the estimation of the correlation matrices. We first evaluate the convergence speed of DANSE$_K$ and S-DANSE$_K$ in a scenario for which S-DANSE$_K$ converges. Then, we demonstrate the convergence of rS-DANSE$_K^+$ and rS-DANSE$_K$ in a scenario for which S-DANSE$_K$ gets locked in a limit cycle.

The node-specific desired signals $\mathbf{d}_k$ are random mixtures of the Q channels of a latent signal $\mathbf{d}$, where $Q = 3$. All three channels of $\mathbf{d}$ are uniformly distributed random processes from which $N = 10000$ samples are generated. The M_k sensor signals of node k in $\mathbf{y}_k$ consist of different random mixtures of the three channels of the latent signal $\mathbf{d}$, with some uncorrelated Gaussian white noise added with half the power of the channels of $\mathbf{d}$. Each node has 10 sensors, i.e., $M_k = 10, \forall\, k \in \mathcal{J}$. All evaluations of the MSE cost functions J_k are performed on the equivalent least-squares (LS) cost functions, i.e.

$$\sum_{t=0}^{N} \|\mathbf{d}_k[t] - \mathbf{W}_k^H \mathbf{y}[t]\|^2. \tag{36}$$

Also, the correlation matrices are replaced by their least squares equivalent, i.e., $E\{\mathbf{y}\mathbf{y}^H\}$ is replaced by $\mathbf{Y}\mathbf{Y}^H$ where $\mathbf{Y}$ denotes an $M \times N$ sample matrix that contains samples of the variable $\mathbf{y}$ in its columns.

The DANSE$_3$ algorithm and the S-DANSE$_3$ algorithm are simulated in networks with $J = 4$, $J = 8$, and $J = 15$ nodes. The result for node 1 is shown in Fig. 4. Notice that this is a case where S-DANSE$_3$ converges, and, therefore, no relaxation is required. Not surprisingly, the convergence time of the DANSE$_3$ algorithm, using sequential round-robin updates, increases linearly with the number of nodes. On the other hand, the convergence time of the S-DANSE$_3$ algorithm is unaffected when the number of nodes is increased. This shows that the simultaneous updating procedure generally yields faster convergence, especially when the number of nodes is large. Notice that in both algorithms, the algorithm has converged once each node has updated three times, irrespective of the number of nodes in the network.

Fig. 5(a) shows a simulation result with $Q = 1$ where S-DANSE$_1$ does not converge to the optimal solution, but gets locked in a limit cycle. The network has $J = 4$ nodes with $M_k = 10$ for all k. All sensor signals are random mixtures of three white noise signals. Differently scaled versions of one of these white noise signals are used as desired signals for all four nodes. The other curves in Fig. 5(a) correspond to 3 versions of the rS-DANSE$_1$ algorithm. In the first version, α^i is set to $\alpha^i = 0.7\ \forall\, i \in \mathbb{N}$, again resulting in a limit cycle. In the second version, α^i is set to $\alpha^i = 0.3\ \forall\, i \in \mathbb{N}$, now yielding convergence to the optimal solution. In the third version, we choose the sequence $(\alpha^i)_{i\in\mathbb{N}}$ equal to $\alpha^i = 1/i$, again yielding convergence.

Fig. 5(b) compares the convergence speed of rS-DANSE$_1$ with rS-DANSE$_1^+$ for the same scenario. Two relaxed versions are tested: $\alpha^i = 0.3\ \forall\, i \in \mathbb{N}$, and $\alpha^i = 1/i$. It is observed that rS-DANSE$_1^+$ converges faster due to the extra update (22). Notice that the version with $\alpha^i = 1/i$ has an initial fast convergence, but eventually becomes very slow due to the decreasing stepsize.

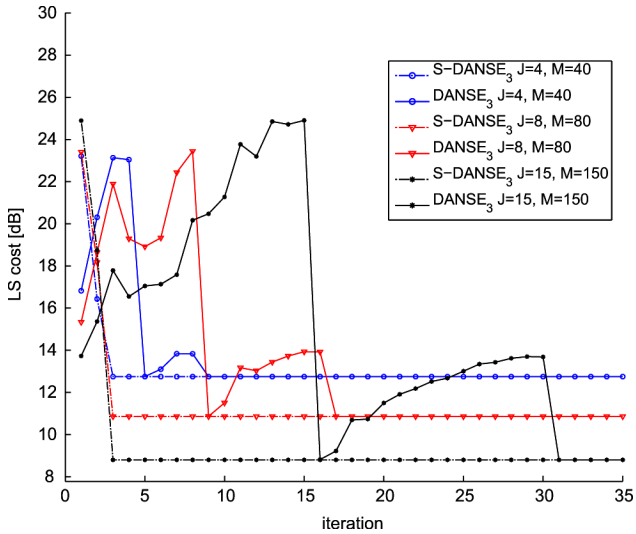

Fig. 4. LS error of node 1 versus iteration i for networks with $J = 4$, $J = 8$ and $J = 15$ nodes respectively.

Simulation results with speech signals are provided in a follow-up paper [12]. In this paper, a distributed speech enhancement algorithm, based on the S-DANSE$_K$ algorithm and its relaxed variations, is tested in a simulated acoustic sensor network scenario. It is observed that limit cycles occur quite frequently, and, therefore, relaxation is required when nodes update simultaneously. Even though relaxation affects the adaptation speed, it is observed that rS-DANSE converges faster than DANSE with a sequential updating procedure.

B. Adaptive Implementation

In this section, we evaluate the tracking performance of the adaptive implementation of the rS-DANSE algorithm. The main difference with the batch mode simulations is that subsequent iterations are now performed on different signal segments, i.e., the same data segment is never used twice. This introduces a tradeoff between tracking performance and estimation performance, which should be taken into account when a window length is chosen for the estimation of the signal statistics. Indeed, to have a fast tracking, the statistics must be estimated from short signal segments, yielding larger estimation errors in the correlation matrices that are used to compute the estimators at the different nodes. Another major difference with Section V-A is that the correlation matrices are now estimated in a specific fashion by exploiting the ON-OFF behavior of the target sources. We refer to [10] for further details.

The scenario is depicted in Fig. 6. The network contains $J = 4$ nodes ($\diamond$). Each node has a reference sensor at the node itself, and can collect observations of 5 additional sensors ($\circ$) that are uniformly distributed within a 1.6 m radius around the node. Eight localized white Gaussian noise sources ($\bigtriangledown$) are present. Two ($Q = 2$) target sources ($\square$) move back and forth over the indicated straight lines, and halt for 2 s at the end points of these lines. The goal for each node is to estimate the desired component d_{k1} in its reference sensor, which is a signal that consists of an unknown mixture of the two target sources, without any noise. Since the simulated scenario is exactly the same as in

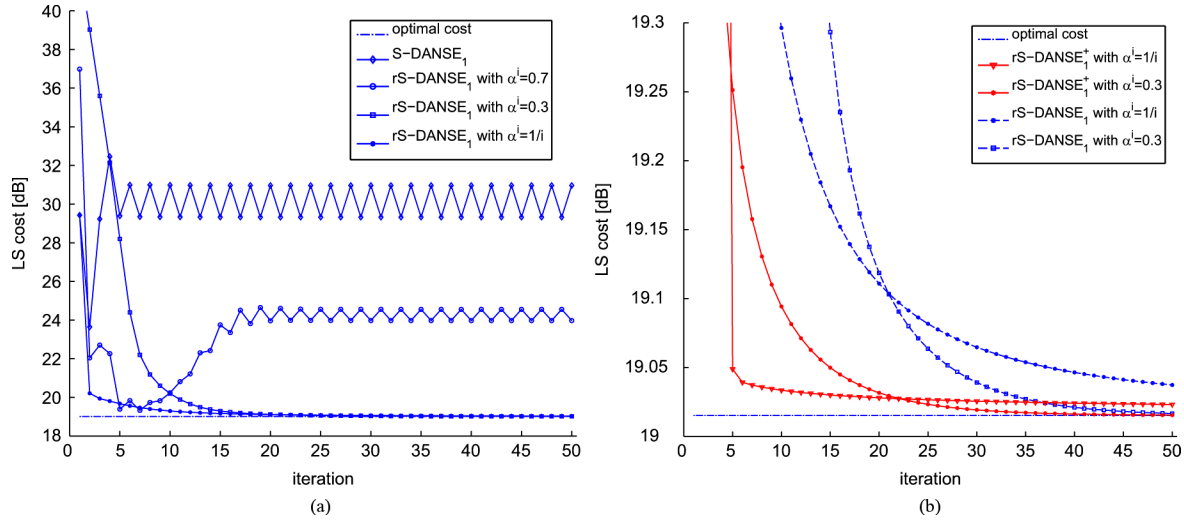

Fig. 5. Log-plot of the LS error of node 1 versus iteration i, for a case in which S-DANSE$_1$ does not converge. The network has $J = 4$ nodes with $M_k = 10$ for all k, and with $K = Q = 1$. By decreasing the relaxation parameter in the rS-DANSE$_1$ or rS-DANSE$_1^+$ algorithm, convergence occurs to the optimal value. (a) S-DANSE$_1$ versus rS-DANSE$_1$. (b) rS-DANSE$_1^+$ versus rS-DANSE$_1$.

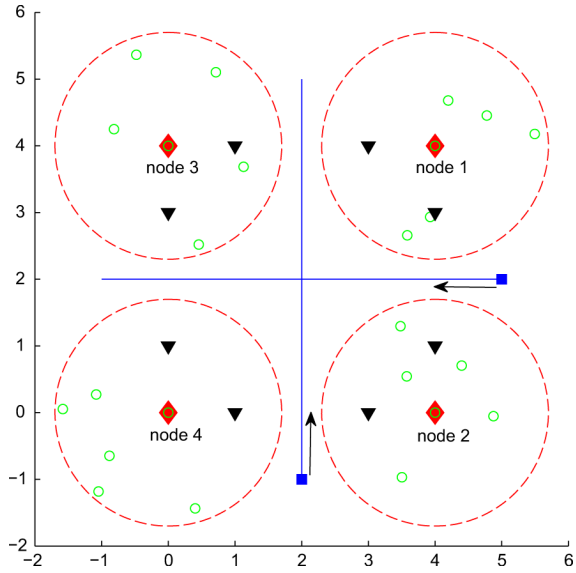

Fig. 6. Description of the simulated scenario. The network contains 4 nodes ($\diamond$), each node collecting observations in a cluster of six sensors ($\circ$). One sensor of each cluster is positioned at the node itself. Two target sources ($\square$) are moving over the indicated straight lines. Eight noise sources are present ($\triangledown$).

[10], we refer to the latter for further details on the data model and the signals involved.

We will use the signal-to-error ratio (SER) as a measure to assess the performance of the estimators. The instantaneous SER for node k at time t and iteration i is computed over 3200 samples, and is defined as

$$\mathrm{SER}_k^i[t] = \frac{\sum_{n=t+1}^{t+3200} |d_{k1}[n]|^2}{\sum_{n=t+1}^{t+3200} |d_{k1}[n] - \tilde{\mathbf{w}}_k^i(1)^H \mathbf{y}[n]|^2} \tag{37}$$

where $\tilde{\mathbf{w}}_k^i(1)$ denotes the first column of the estimator $\tilde{\mathbf{W}}_k^i$, as defined in (7). Notice that this is the estimator that is of actual interest, since it estimates the desired component d_{k1} in the reference sensor.

Fig. 7 shows the SER over time in the outputs of the four nodes, both for DANSE$_2$ and rS-DANSE$_2$, where the latter uses the relaxation parameter[10] $\alpha^i = 0.5, \forall\, i \in \mathbb{N}$, for all four nodes. The window lengths for the estimation of the sensor signal correlation matrices and the noise correlation matrices[11] are $L_1 = 4200$ and $L_2 = 2200$, respectively. The time between two consecutive updates is 0.4 s, i.e., the iteration index i changes to $i + 1$ every 0.4 s. The sources move at a speed of 1 m/s. Dashed vertical lines are plotted to indicate the points in time where both sources start moving, and full vertical lines indicate when they halt. The sources stand still in the time intervals [0–4] s, [10–12] s and [18–20] s. The performance is compared to the performance of the centralized version, in which all sensor signals are centralized in a single fusion center that computes the optimal estimators according to (4). The centralized version updates its estimators at every sample time, which results in very fast convergence.

It is observed that the rS-DANSE$_2$ algorithm has better tracking performance than the DANSE$_2$ algorithm with sequential updating. The latter has dips in the SER, corresponding to the time it takes to update all the nodes to obtain a good estimator for the current position of the sources. The rS-DANSE$_2$ can react more swiftly to the changing positions of the target sources, and is more or less able to follow the centralized estimators.

The same experiment is performed with the target sources moving three times faster, i.e., at a speed of 3 m/s. Now, the sources stand still in the time intervals [0–4] s, [6–8] s, and [10–12] s. The results are shown in Fig. 8. It is observed

[10]The value of the relaxation parameter is chosen manually. The S-DANSE$_2$ algorithm without relaxation is observed not to converge in this adaptive experiment.

[11]The noise correlation matrix is a sensor signal correlation matrix that is computed during noise-only periods, i.e., when the target sources are OFF. We refer to [10] for further details on why this noise correlation matrix has to be computed.

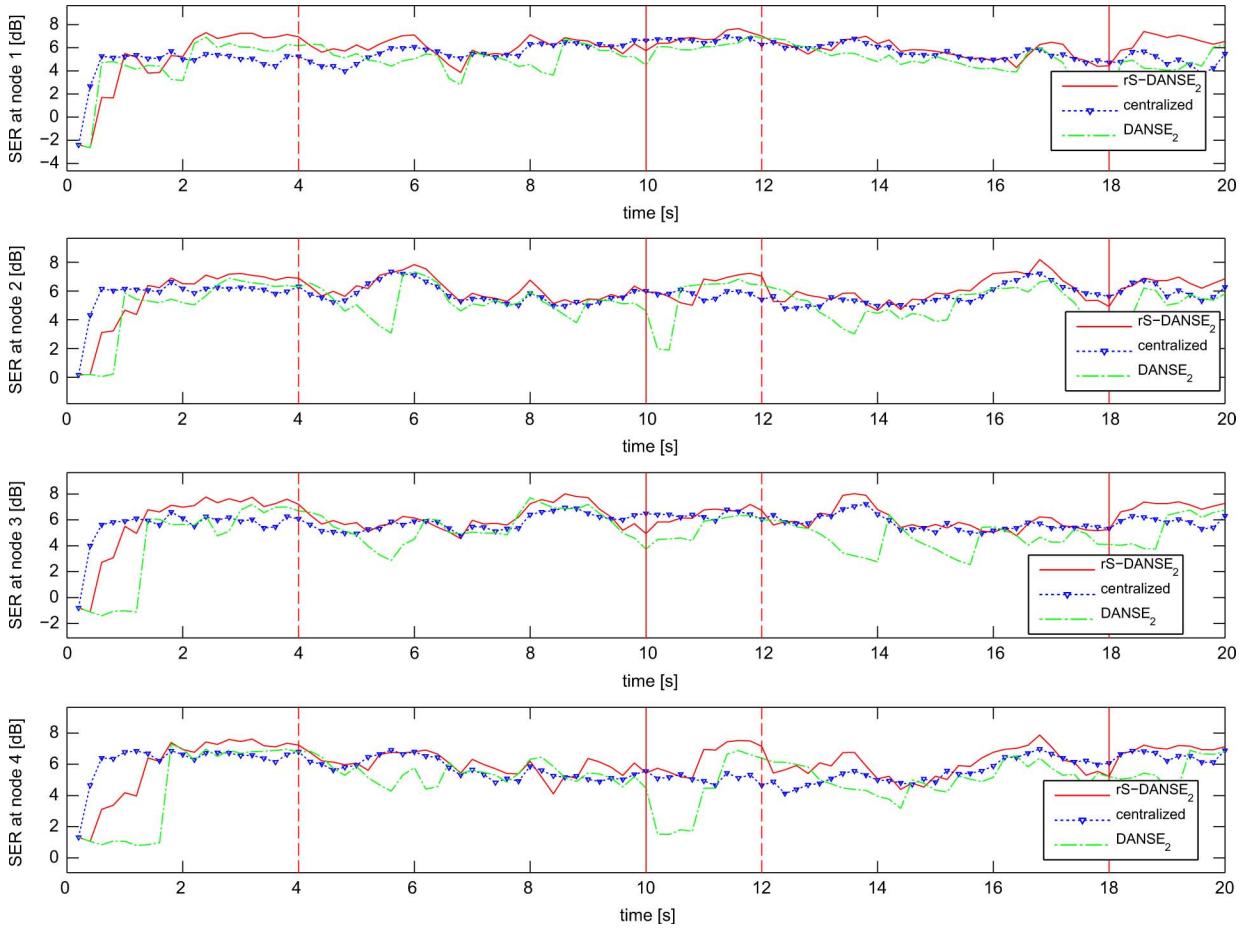

Fig. 7. SER of DANSE_2 and rS-DANSE_2 versus time at the 4 nodes depicted in Fig. 6, when the sources move at 1 m/s. The centralized version is added as a reference.

that rS-DANSE_2 now has more difficulty following the centralized algorithm. However, once the sources stand still, the rS-DANSE_2 algorithm always regains optimality. The difference between rS-DANSE_2 and DANSE_2 is now even more significant, especially at nodes 3 and 4, where DANSE_2 shows some large SER dips.

VI. Conclusion

In this paper, we have investigated the case in which the nodes in the DANSE_K algorithm perform their parameter updates simultaneously or asynchronously. We have pointed out that the convergence and optimality of the DANSE_K algorithm can no longer be guaranteed in this case. We have then modified the DANSE_K algorithm with a relaxation operation to restore convergence and optimality under simultaneous and asynchronous updating. Convergence of the new algorithm is proven if the relaxation parameter satisfies certain conditions. Simulations indicate that a simplified version of the new algorithm can also be used, with a lower computational load, while maintaining convergence. Since all nodes can estimate the required statistics in parallel, the new algorithm adapts faster than the original DANSE_K algorithm, especially so when the number of nodes is large. The convergence results are demonstrated with simulations in batch mode as well as with an adaptive version of the algorithm.

Appendix

Proof of Theorem IV.1:

For the sake of an easy exposition, we assume here that all signals are real valued. As explained in Appendix B, the case with complex valued signals can be transformed to a real valued case for which the proof infra immediately applies. In the proof, we also assume that all signal statistics are perfectly known, i.e., it is as if the algorithm uses an infinite observation window.

We define the cost function

$$J_k^p(\mathbf{w}_k) = |d_k(p) - \mathbf{w}_k^H \mathbf{y}|^2 \tag{38}$$

where $d_k(p)$ denotes the pth entry of the vector $\mathbf{d}_k$. The cost function of the MMSE problem (3) is then equal to

$$J_k(\mathbf{W}_k) = \sum_{p=1}^{K} J_k^p(\mathbf{w}_k(p)) \tag{39}$$

where $\mathbf{w}_k(p)$ denotes the pth column of $\mathbf{W}_k$. Similarly, we define $\mathbf{g}_k(p)$ as the pth column of $\mathbf{G}_k$, and

$$\bar{\mathbf{W}}_k^i = \begin{bmatrix} \mathbf{W}_{11}^i \mathbf{G}_{k1}^{i+1} \\ \vdots \\ \mathbf{W}_{JJ}^i \mathbf{G}_{kJ}^{i+1} \end{bmatrix} \tag{40}$$

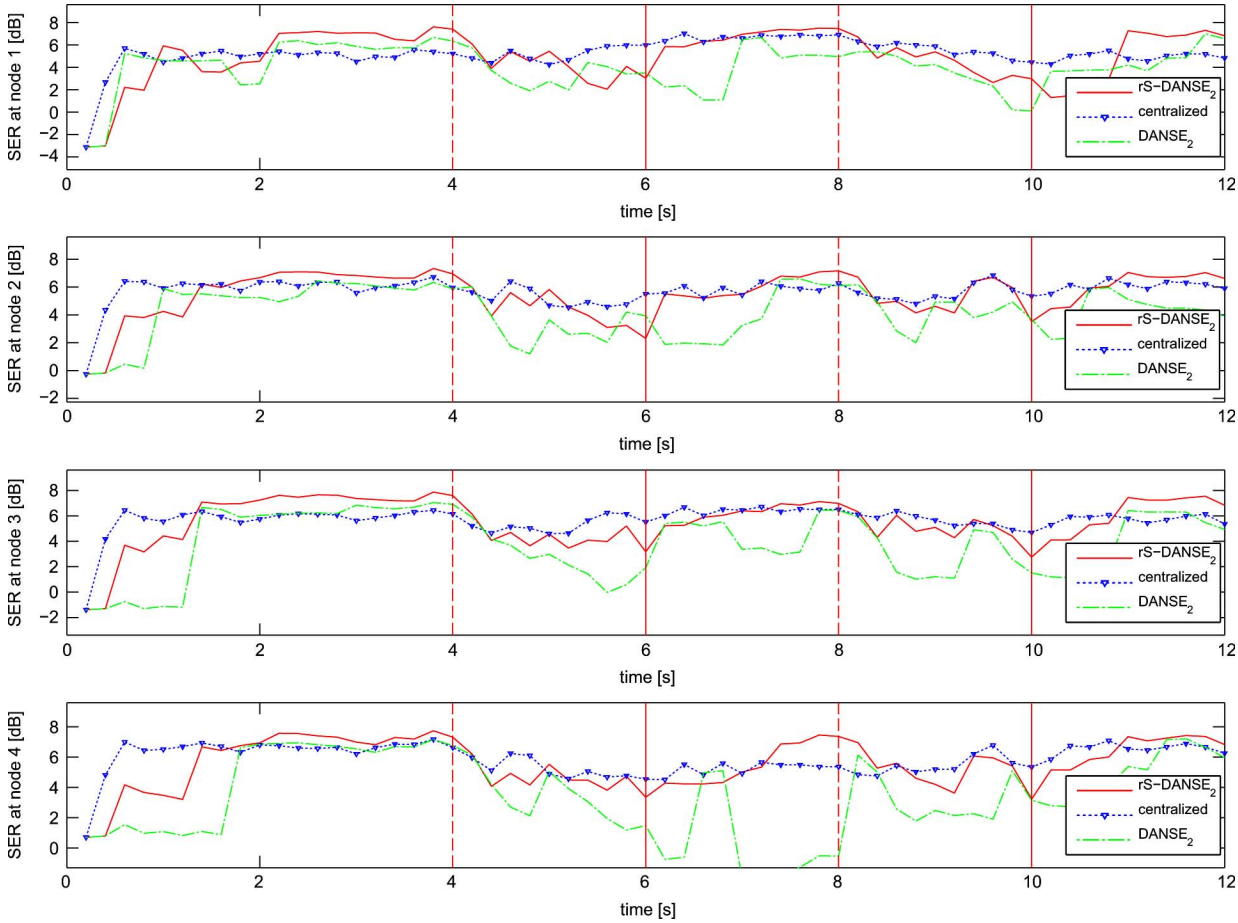

Fig. 8. SER of DANSE_2 and rS-DANSE_2 versus time at the 4 nodes depicted in Fig. 6, when the sources move at 3 m/s. The centralized version is added as a reference.

where $\mathbf{G}_k^{i+1} = [\mathbf{G}_{k1}^{i+1\,T} \dots \mathbf{G}_{kJ}^{i+1\,T}]^T$ is computed according to (23).

From (1), it readily follows that $\mathbf{d}_k^H = \mathbf{d}_q^H \mathbf{A}_{kq}$, where $\mathbf{A}_{kq} = \mathbf{A}_q^{-H}\mathbf{A}_k$. Using this, we can show that the results of update (23) are the same for all k, up to a transformation with the $K \times K$ matrix $\mathbf{A}_{kq}$, and therefore

$$\bar{\mathbf{W}}_k^i = \bar{\mathbf{W}}_q^i \mathbf{A}_{kq}. \tag{41}$$

We define the M-dimensional vector

$$\mathbf{s}_k^i(p) = \left[\mathbf{o}_{M_1}^T \cdots \mathbf{o}_{M_{k-1}}^T \; \mathbf{w}_{kk}^{iT}(p) \; \mathbf{o}_{M_{k+1}}^T \cdots \mathbf{o}_{M_J}^T\right]^T \tag{42}$$

where $\mathbf{o}_{M_k}$ denotes an all-zero M_k-dimensional vector, and where $\mathbf{w}_{kk}^i(p)$ denotes the pth column of $\mathbf{W}_{kk}^i$. We define the following relationship between the matrix $\mathbf{W}^i$ and a KJ-dimensional subspace S^i of $\mathbb{R}^M$:

$$\begin{aligned}\mathbf{W}^i \rightarrow S^i \\ = Span\{\mathbf{s}_1^i(1), \dots, \mathbf{s}_1^i(K), \dots, \mathbf{s}_J^i(1), \dots, \mathbf{s}_J^i(K)\}.\end{aligned} \tag{43}$$

By definition, any column of $\mathbf{W}^i$ is inside S^i. The same holds true for all M-dimensional vectors of the form

$$\begin{bmatrix} \mathbf{W}_{11}^i \mathbf{h}_1 \\ \vdots \\ \mathbf{W}_{JJ}^i \mathbf{h}_J \end{bmatrix} \tag{44}$$

where $\mathbf{h}_q \in \mathbb{R}^K$, $\forall\, q \in \mathcal{J}$. Therefore, after update (23), the following expression holds:

$$J_k^p(\bar{\mathbf{W}}_k^i(p)) = \min_{\mathbf{w}_k \in S^i} J_k^p(\mathbf{w}_k) \tag{45}$$

i.e., (23) defines the solution of the constrained optimization problem defined by the RHS of (45), $\forall\, p \in \{1, \dots, K\}$, where S^i defines the search space. This means that the subspace S^i is tangent to a sublevel set of J_k^p in the point $\bar{\mathbf{W}}_k^i(p)$, i.e., the pth column of $\bar{\mathbf{W}}_k^i$. Therefore, the gradient ∇J_k^p is orthogonal to S^i in $\bar{\mathbf{W}}_k^i(p)$, and, therefore, orthogonal to all vectors $\mathbf{s}_k^i(l)$, $\forall\, k \in \mathcal{J}$, $\forall\, l \in \{1, \dots, K\}$. With (42), it is clear that the partitions of the gradient, therefore, satisfy

$$\forall k, q \in \mathcal{J}, \forall p, l \in \{1, \dots, K\} : \nabla_{\mathbf{w}_{kq}} J_k^p \left(\bar{\mathbf{W}}_k^i(p)\right) \perp \mathbf{w}_{qq}^i(l). \tag{46}$$

Now assume hypothetically, and without loss of generality, that node 1 updates the current values $\mathbf{W}_{11}^i$ and $\mathbf{G}_1^{i+1}$ to $\mathbf{W}_{11}^{(i+1)}$ and $\mathbf{G}_1^{(i+1)}$, respectively, following update (10) of the DANSE_K algorithm. We added brackets to the iteration index to avoid confusion with update (22)–(28). From the proof of Theorem III.1, as given in [10], it is found that [see (47) at

the bottom of the page]. We can show that this expression also holds for every cost function J_k^p separately. Therefore, the inequality holds [see (48), shown at the bottom of the page], where $\mathbf{a}_{k1}(p)$ denotes the pth column of $\mathbf{A}_{k1}$. By using (48), and since the cost functions are strictly convex, the directional vector $\triangle\mathbf{w}_k^{(i+1)}(p)$, defined by

$$\triangle\mathbf{w}_k^{(i+1)}(p) = \begin{bmatrix} \mathbf{W}_{11}^{(i+1)}\mathbf{a}_{k1}(p) - \mathbf{W}_{11}^{i}\mathbf{g}_{k1}^{i+1}(p) \\ \mathbf{W}_{22}^{i}(\mathbf{G}_{12}^{(i+1)}\mathbf{a}_{k1}(p) - \mathbf{g}_{k2}^{i+1}(p)) \\ \vdots \\ \mathbf{W}_{JJ}^{i}(\mathbf{G}_{1J}^{(i+1)}\mathbf{a}_{k1}(p) - \mathbf{g}_{kJ}^{i+1}(p)) \end{bmatrix} \tag{49}$$

defines a nonincreasing direction of J_k^p in the current point $\bar{\mathbf{W}}_k^i(p)$. This means that

$$\forall\, k \in \mathcal{J},\ \forall\, p \in \{1,\ldots,K\} : \nabla J_k^p\left(\bar{\mathbf{W}}_k^i(p)\right)^T \triangle\mathbf{w}_k^{(i+1)}(p) \leq 0. \tag{50}$$

With (46), and the fact that $\mathbf{W}_{11}^{(i+1)} = F_1(\mathbf{W}^i)$, this yields (51) at the bottom of the page. By using a similar reasoning as above for any hypothetical update of any node q, we can show that [see (52) at the bottom of the page] or equivalently, when using the matrix-valued gradient of the global cost function $J_k(\mathbf{W}_k)$.

$$\forall\, k \in \mathcal{J},\ \forall\, q \in \mathcal{J} : \mathrm{tr}\left(\nabla_{\mathbf{W_{kq}}} J_k\left(\bar{\mathbf{W}}_k^i\right)^T F_q\left(\mathbf{W}^i\right)\mathbf{A}_{kq}\right) \leq 0 \tag{53}$$

with $\mathrm{tr}(\mathbf{A})$ denoting the trace of matrix $\mathbf{A}$. It is noted that there is at least one $q \in \mathcal{J}$ for which the inequality in (53) is a strict inequality, as long as the equilibrium point has not been reached, i.e., as long as $\bar{\mathbf{W}}_k^i \neq \hat{\mathbf{W}}_k$, where $\hat{\mathbf{W}}_k$ denotes the optimal solution (4). Indeed, due to the strict convexity of the cost function J_k, the inequality in (48) is strict for at least one $p \in \{1,\ldots,J\}$ and at least one hypothetically updating node $q \in \mathcal{J}$ (although it might be required to choose another hypothetically updating node instead of node 1, which was chosen w.l.o.g. in (48)). Therefore, we find (54) at the bottom of the page. By defining

$$\triangle\mathbf{W}_{kk}^{i+1} = F_k(\mathbf{W}^i) - \bar{\mathbf{W}}_{kk}^i \tag{55}$$

$$\forall\, k \in \mathcal{J},\ \forall\, \mathbf{H}_k \in \mathbb{R}^{K\times K} : J_k\left(\begin{bmatrix} \mathbf{W}_{11}^{(i+1)} \\ \mathbf{W}_{22}^{i}\mathbf{G}_{12}^{(i+1)} \\ \vdots \\ \mathbf{W}_{JJ}^{i}\mathbf{G}_{1J}^{(i+1)} \end{bmatrix}\mathbf{A}_{k1}\right) \leq J_k\left(\begin{bmatrix} \mathbf{W}_{11}^{i}\mathbf{H}_1 \\ \vdots \\ \mathbf{W}_{JJ}^{i}\mathbf{H}_J \end{bmatrix}\right). \tag{47}$$

$$\forall\, k \in \mathcal{J},\ \forall\, p \in \{1\ldots K\} : J_k^p\left(\begin{bmatrix} \mathbf{W}_{11}^{(i+1)} \\ \mathbf{W}_{22}^{i}\mathbf{G}_{12}^{(i+1)} \\ \vdots \\ \mathbf{W}_{JJ}^{i}\mathbf{G}_{1J}^{(i+1)} \end{bmatrix}\mathbf{a}_{k1}(p)\right) \leq J_k^p\left(\begin{bmatrix} \mathbf{W}_{11}^{i}\mathbf{g}_{k1}^{i+1}(p) \\ \vdots \\ \mathbf{W}_{JJ}^{i}\mathbf{g}_{kJ}^{i+1}(p) \end{bmatrix}\right) \tag{48}$$

$$\forall\, k \in \mathcal{J},\ \forall\, p \in \{1,\ldots,K\} : \sum_{p=1}^{K}\left(\nabla_{\mathbf{w}_{k1}} J_k^p\left(\bar{\mathbf{W}}_k^i(p)\right)^T F_1(\mathbf{W}^i)\mathbf{a}_{k1}(p)\right) \leq 0. \tag{51}$$

$$\forall\, k,q \in \mathcal{J},\ \forall\, p \in \{1,\ldots,K\} : \sum_{p=1}^{K}\left(\nabla_{\mathbf{w}_{kq}} J_k^p\left(\bar{\mathbf{W}}_k^i(p)\right)^T F_q(\mathbf{W}^i)\mathbf{a}_{kq}(p)\right) \leq 0 \tag{52}$$

$$\bar{\mathbf{W}}_k^i \neq \hat{\mathbf{W}}_k \Leftrightarrow \forall\, k \in \mathcal{J},\ \exists\, q \in \mathcal{J} : \mathrm{tr}\left(\nabla_{\mathbf{W_{kq}}} J_k\left(\bar{\mathbf{W}}_k^i\right)^T F_q\left(\mathbf{W}^i\right)\mathbf{A}_{kq}\right) < 0. \tag{54}$$

we can rewrite (24) as

$$\mathbf{W}_{kk}^{i+1} = \bar{\mathbf{W}}_{kk}^{i} + \alpha^i \triangle \mathbf{W}_{kk}^{i+1}. \tag{56}$$

With this, the following expression is obtained:

$$J_k(\bar{\mathbf{W}}_k^{i+1}) = \min_{\mathbf{G}_k} J_k\left(\begin{bmatrix} \mathbf{W}_{11}^{i+1}\mathbf{G}_{k1} \\ \vdots \\ \mathbf{W}_{JJ}^{i+1}\mathbf{G}_{kJ} \end{bmatrix}\right) \tag{57}$$

$$\leq J_k\left(\begin{bmatrix} \left(\bar{\mathbf{W}}_{11}^{i} + \alpha^i \triangle \mathbf{W}_{11}^{i+1}\right)\mathbf{A}_{k1} \\ \vdots \\ \left(\bar{\mathbf{W}}_{JJ}^{i} + \alpha^i \triangle \mathbf{W}_{JJ}^{i+1}\right)\mathbf{A}_{kJ} \end{bmatrix}\right). \tag{58}$$

With (41), the RHS of (58) is equal to

$$J_k\left(\bar{\mathbf{W}}_k^{i} + \alpha^i \begin{bmatrix} \triangle\mathbf{W}_{11}^{i+1}\mathbf{A}_{k1} \\ \vdots \\ \triangle\mathbf{W}_{JJ}^{i+1}\mathbf{A}_{kJ} \end{bmatrix}\right). \tag{59}$$

By using a first order Taylor expansion of the function J_k in the point $\bar{\mathbf{W}}_k^i$, we can rewrite (59) as

$$J_k\left(\bar{\mathbf{W}}_k^{i}\right) + \alpha^i \mathrm{tr}\left(\nabla J_k\left(\bar{\mathbf{W}}_k^{i}\right)^T \begin{bmatrix} \triangle\mathbf{W}_{11}^{i+1}\mathbf{A}_{k1} \\ \vdots \\ \triangle\mathbf{W}_{JJ}^{i+1}\mathbf{A}_{kJ} \end{bmatrix}\right) + \epsilon(\bar{\mathbf{W}}_k^i, \alpha^i) \tag{60}$$

with the error term satisfying

$$\lim_{\alpha^i \longrightarrow 0} \frac{\epsilon(\bar{\mathbf{W}}_k^i, \alpha^i)}{\alpha^i} = 0. \tag{61}$$

The error term only depends on $\bar{\mathbf{W}}_k^i$ and α^i, i.e., the difference vector is implicitly computed from these two variables. With (55) and (46), we can rewrite (60) as (62), shown at the bottom of the page. We can condense the chain (58)–(62) to

$$J_k\left(\bar{\mathbf{W}}_k^{i+1}\right) \leq J_k\left(\bar{\mathbf{W}}_k^{i}\right) + \alpha^i\left(V_k(\bar{\mathbf{W}}_k^i) + \frac{\epsilon(\bar{\mathbf{W}}_k^i, \alpha^i)}{\alpha^i}\right) \tag{63}$$

where $V_k(\bar{\mathbf{W}}_k^i)$ denotes the function[12] described by the second term of (62). Because of (53) and (54), we know that

$$V_k(\bar{\mathbf{W}}_k^i) < 0 \Leftrightarrow \bar{\mathbf{W}}_k^i \neq \hat{\mathbf{W}}_k. \tag{64}$$

Based on this observation, we will prove that the sequence $\left(\tilde{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ converges to $\hat{\mathbf{W}}_k$. Since convergence of $\left(\tilde{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ to $\hat{\mathbf{W}}_k$ is equivalent to convergence of the sequence $\left(\bar{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ to $\hat{\mathbf{W}}_k$, we only prove the latter. We do this by proving that

[12]Since $F(\bar{\mathbf{W}}_k^i) = F(\mathbf{W}^i)$, only $\bar{\mathbf{W}}_k^i$ is explicitly stated in the argument of the function V_k.

$$\forall\, \delta \in \mathbb{R}_0^+, \exists\, L \in \mathbb{N} : i > L \Rightarrow \|\hat{\mathbf{W}}_k - \bar{\mathbf{W}}_k^i\| < \delta. \tag{65}$$

Let $B_1 = B\left(\hat{\mathbf{W}}_k, \delta\right)$ denote a ball with center point $\hat{\mathbf{W}}_k$ and radius δ that is open, i.e., it does not contain its boundary. Since J_k is a convex quadratic function, there exists a sublevel set C of J_k that satisfies $C \subseteq B_1$. We choose any open ball $B_2 = B\left(\hat{\mathbf{W}}_k, \delta_2\right) \subset C$.

We now show by contradiction that the sequence $\left(\bar{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ contains an infinite number of elements in B_2. Let C_0 be the sublevel set of J_k that satisfies

$$C_0 = \{\mathbf{W}_k | J_k(\mathbf{W}_k) \leq J_k(\mathbf{O}_{M\times K})\} \tag{66}$$

where $\mathbf{O}_{M\times K}$ denotes an $M \times K$ all-zero matrix. Because of (22), $\bar{\mathbf{W}}_k^i$ is always inside C_0. We define

$$\beta = \max_{\mathbf{W}_k \in C_0 \setminus B_2} V_k(\mathbf{W}_k). \tag{67}$$

Because of (64) and the fact that B_2 is an open set, we know that this maximum exists, and that $\beta < 0$. We define

$$\epsilon^*(\alpha^i) = \max_{\mathbf{W}_k \in C_0 \setminus B_2} \epsilon\left(\mathbf{W}_k, \alpha^i\right). \tag{68}$$

Because of (27) and (61), we know that

$$\exists L_1 \in \mathbb{N} : i > L_1 \Rightarrow \frac{\epsilon^*(\alpha^i)}{\alpha^i} < |\beta| - \kappa \tag{69}$$

with $\kappa \in (0, |\beta|)$. Let

$$\gamma = \sup\left\{\frac{\epsilon^*(\alpha^i)}{\alpha^i} | i > L_1\right\}, \tag{70}$$

and

$$\phi = \beta + \gamma. \tag{71}$$

By construction, ϕ must be strictly negative. By using (67)–(70), it follows from (63) that

$$(i > L_1 \wedge \bar{\mathbf{W}}_k^i \notin B_2) \Rightarrow J_k\left(\bar{\mathbf{W}}_k^{i+1}\right) \leq J_k\left(\bar{\mathbf{W}}_k^{i}\right) + \alpha^i\phi \tag{72}$$

with $\wedge$ denoting the logic AND operator. Now suppose that $\exists\, L_2 > L_1 : i > L_2 \Rightarrow \bar{\mathbf{W}}_k^i \notin B_2$, then, because of (28) and the fact that ϕ is strictly negative

$$\lim_{i \longrightarrow \infty} J_k\left(\bar{\mathbf{W}}_k^i\right) = -\infty. \tag{73}$$

However, J_k has a lower bound, and, therefore, no such L_2 exists. This shows that there are an infinite number of elements in $\left(\bar{\mathbf{W}}_k^i\right)_{i\in\mathbb{N}}$ that are inside B_2.

$$J_k\left(\bar{\mathbf{W}}_k^{i}\right) + \alpha^i \mathrm{tr}\left(\nabla J_k\left(\bar{\mathbf{W}}_k^{i}\right)^T \begin{bmatrix} F_1\left(\mathbf{W}^i\right)\mathbf{A}_{k1} \\ \vdots \\ F_J\left(\mathbf{W}^i\right)\mathbf{A}_{kJ} \end{bmatrix}\right) + \epsilon(\bar{\mathbf{W}}_k^i, \alpha^i). \tag{62}$$

Because of (27), the following expression holds:

$$\begin{gathered}\exists L_3 \in \mathbb{N} : i > L_3 \\ \Downarrow \\ \max_{\mathbf{W}_k \in B_2} \|Z_k^i(\mathbf{W}_k) - \mathbf{W}_k\| < \min_{\mathbf{U}_1 \in \partial C,\, \mathbf{U}_2 \in B_2} \|\mathbf{U}_1 - \mathbf{U}_2\|\end{gathered} \tag{74}$$

where ∂C is the boundary of the sublevel set C, and where $Z_k^i(\mathbf{W}_k)$ is defined as the function such that $\bar{\mathbf{W}}_k^{i+1} = Z_k^i(\mathbf{W}_k^i)$ according to the parallel update rules (22)–(28). Choose $L_4 > \max(L_1, L_3)$. Since there are an infinite number of elements in $\left(\bar{\mathbf{W}}_k^i\right)_{i \in \mathbb{N}}$ that are inside B_2, there exists an $i^* > L_4$ such that $\bar{\mathbf{W}}_k^{i^*}$ is inside B_2. Now we prove the induction hypothesis

$$\forall\, i > L_4 : \bar{\mathbf{W}}_k^i \in C \Rightarrow \bar{\mathbf{W}}_k^{i+1} \in C. \tag{75}$$

Assume that $\bar{\mathbf{W}}_k^i \in C$, then either $\bar{\mathbf{W}}_k^i \in C \backslash B_2$ or $\bar{\mathbf{W}}_k^i \in B_2$. In the case where $\bar{\mathbf{W}}_k^i \in C \backslash B_2$, we know that $\bar{\mathbf{W}}_k^{i+1} \in C$ because of (72) and the fact that $\phi < 0$. If $\bar{\mathbf{W}}_k^i \in B_2$, then $\bar{\mathbf{W}}_k^{i+1} \in C$ because of (74).

By applying the induction hypothesis (75) to $\bar{\mathbf{W}}_k^{i^*}$, we find that

$$\forall\, i > i^* : \bar{\mathbf{W}}_k^i \in C. \tag{76}$$

Because $C \subset B_1$, we have proved that (65) holds when choosing $L = i^*$. This completes the proof. ∎

Transformation of Complex Valued DANSE to Real Valued DANSE: In this section, we briefly show how the DANSE scheme with complex valued signals can be transformed to an isomorph scheme with only real valued signals. We define the following isomorphism between a complex number z and the 2×2 real matrix $\check{z}$:

$$z \leftrightarrow \check{z} = \begin{bmatrix} \Re z & -\Im z \\ \Im z & \Re z \end{bmatrix} \tag{77}$$

where $\Re z$ and $\Im z$ denote the real and the imaginary part of z. We also define $\grave{z}$ and $\acute{z}$ as the first and second column of $\check{z}$. The isomorphism (77) defines another isomorphism between matrices

$$\mathbf{A} \in \mathbb{C}^{m \times n} \leftrightarrow \check{\mathbf{A}} \in \mathbb{R}^{2m \times 2n} \tag{78}$$

in such a way that each entry A_{ij} in $\mathbf{A}$ is replaced by the 2×2 matrix $\check{A}_{ij}$. Similarly, we define $\grave{\mathbf{A}}$ and $\acute{\mathbf{A}}$ as the matrices in which A_{ij} is replaced by $\grave{A}_{ij}$ and $\acute{A}_{ij}$, respectively. Notice that

$$z^* \leftrightarrow \check{z}^T \tag{79}$$

$$\mathbf{A}^H \leftrightarrow \check{\mathbf{A}}^T. \tag{80}$$

The cost function $J_k(\mathbf{W}_k)$, defined in (17), can now be transformed[13] to

$$\check{J}_k(\grave{\mathbf{W}}_k) = \frac{1}{2} E\left\{\|\grave{\mathbf{d}}_k - \grave{\mathbf{W}}_k^T \grave{\mathbf{y}}\|_F^2\right\} \tag{81}$$

with $\|.\|_F$ denoting the Frobenius norm. This cost function has real valued variables, and satisfies $\check{J}_k(\grave{\mathbf{W}}_k) = J_k(\mathbf{W}_k)$. The cost functions $\check{J}_k^p$ are defined similarly. If we let $\mathbf{D}$ denote the gradient of J_k^p in a certain point $\mathbf{w}_k$, then

$$\nabla \check{J}_k^p\left(\grave{\mathbf{w}}_k\right) = \grave{\mathbf{D}}. \tag{82}$$

The proof of Theorem IV.1 can now be extended to the complex valued case, by applying some minor changes using the above isomorphisms. The main changes that have to be made are as follows:

- Apply the isomorphisms (77) and (78) whenever a complex number or matrix is involved. For instance, (7) becomes

$$\check{\grave{\mathbf{W}}}_k^{\,i} = \begin{bmatrix} \check{\mathbf{W}}_{11}^i \check{\mathbf{G}}_{k1}^i \\ \vdots \\ \check{\mathbf{W}}_{JJ}^i \check{\mathbf{G}}_{kJ}^i \end{bmatrix}. \tag{83}$$

- Change all cost functions into their equivalent with real valued variables, similarly to (81).
- The KJ dimensional subspace S^i in (43) now becomes the $2KJ$-dimensional subspace [see (84) at the bottom of the page].
- All inner products between a gradient-vector or -matrix $\mathbf{D}$ and a second vector or matrix $\mathbf{V}$ are applied according to $\grave{\mathbf{D}}^T \grave{\mathbf{V}}$. In this case, (50) and similar expressions remain valid because of (82), and the fact that $\triangle \grave{\mathbf{w}}_k(p)$ is a descent direction of $\check{J}_k^p$ if and only if $\triangle \mathbf{w}_k(p)$ is a descent direction of J_k^p.

REFERENCES

[1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP '01)* , 2001, vol. 4, pp. 2033–2036.

[2] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Process.*, vol. 55, pp. 4064–4077, Aug. 2007.

[3] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, pp. 3122–3136, Jul. 2008.

[4] F. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 56, pp. 1865–1877, May 2008.

[13] We deliberately use (81) instead of $E\left\{\|\grave{\mathbf{d}}_k - \grave{\mathbf{W}}_k^T \grave{\mathbf{y}}\|^2\right\}$, because this makes the relation between J_k and $\check{J}_k$ easier. The reason is that in (81), the full matrix $\grave{\mathbf{W}}_k$ can be treated as a variable in the derivation of the gradient, instead of the argument $\grave{\mathbf{W}}_k$. The resulting gradient $\grave{\mathbf{D}} = 2\mathbf{R}_{\grave{y}\grave{y}} \grave{\mathbf{W}}_k - 2\mathbf{r}_{\grave{y}\grave{d}_k}$ will automatically have the correct structure, i.e., a matrix that is isomorph to the equivalent complex gradient $\mathbf{D} = 2\mathbf{R}_{yy}\mathbf{W}_k - 2\mathbf{r}_{yd_k}$, according to (78). This also holds in other cases, such as the update of $\check{\mathbf{G}}^{i+1}$ according to (23).

$$S_i = Span\{\grave{\mathbf{s}}_1^i(1), \acute{\mathbf{s}}_1^i(1), \ldots, \grave{\mathbf{s}}_1^i(K), \acute{\mathbf{s}}_1^i(K), \ldots, \grave{\mathbf{s}}_J^i(1), \acute{\mathbf{s}}_J^i(1), \ldots, \grave{\mathbf{s}}_J^i(K), \acute{\mathbf{s}}_J^i(K)\}. \tag{84}$$

[5] I. Schizas, G. Giannakis, and Z.-Q. Luo, "Distributed estimation using reduced-dimensionality sensor observations," *IEEE Trans. Signal Process.*, vol. 55, pp. 4284–4299, Aug. 2007.
[6] Z.-Q. Luo, G. Giannakis, and S. Zhang, "Optimal linear decentralized estimation in a bandwidth constrained sensor network," in *Proc. Inf. Theory (ISIT 2005) Int. Symp.*, Sep. 2005, pp. 1441–1445.
[7] K. Zhang, X. Li, P. Zhang, and H. Li, "Optimal linear estimation fusion—Part VI: Sensor data compression," in *Proc. 6th Int. Conf. Inf. Fusion*, 2003, vol. 1, pp. 221–228.
[8] Y. Zhu, E. Song, J. Zhou, and Z. You, "Optimal dimensionality reduction of sensor data in multisensor estimation fusion," *IEEE Trans. Signal Process.*, vol. 53, pp. 1631–1639, May 2005.
[9] A. Bertrand and M. Moonen, "Distributed adaptive estimation of correlated node-specific signals in a fully connected sensor network," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Apr. 2009, pp. 2053–2056.
[10] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks—Part I: Sequential node updating," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5277–5291, Oct. 2010.
[11] S. Doclo, T. van den Bogaert, M. Moonen, and J. Wouters, "Reduced-bandwidth and distributed MWF-based noise reduction algorithms for binaural hearing aids," *IEEE Trans. Audio, Speech Language Process.*, vol. 17, pp. 38–51, Jan. 2009.
[12] A. Bertrand and M. Moonen, "Robust distributed noise reduction in hearing aids with external acoustic sensor nodes," *EURASIP J. Adv. Signal Process.*, vol. 2009, p. 14, 2009.
[13] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.
[14] A. Spriet, M. Moonen, and J. Wouters, "The impact of speech detection errors on the noise reduction performance of multi-channel wiener filtering and generalized sidelobe cancellation," *Signal Process.*, vol. 85, pp. 1073–1088, Jun. 2005.
[15] T. Basar, "Relaxation techniques and asynchronous algorithms for on-line computation of non-cooperative equilibria," *J. Econom. Dyn. Contr.*, vol. 11, pp. 531–549, Dec. 1987.
[16] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA: Athena Scientific, 1997.
[17] S. Uryas'ev and R. Rubinstein, "On relaxation algorithms in computation of noncooperative equilibria," *IEEE Trans. Autom. Control*, vol. 39, pp. 1263–1267, Jun. 1994.



Alexander Bertrand (S'08) was born in Roeselare, Belgium, in 1984. He received the M.Sc. degree in electrical engineering from Katholieke Universiteit Leuven, Belgium, in 2007.

He is currently pursuing the Ph.D. degree with the Electrical Engineering Department (ESAT), Katholieke Universiteit Leuven, and was supported by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). His research interests are in multichannel signal processing, *ad hoc* sensor arrays, wireless sensor networks, distributed signal enhancement, speech enhancement, and distributed estimation.



Marc Moonen (M'94–SM'06–F'07) received the electrical engineering degree and the Ph.D. degree in applied sciences from Katholieke Universiteit Leuven, Belgium, in 1986 and 1990, respectively.

Since 2004, he has been a Full Professor with the Electrical Engineering Department, Katholieke Universiteit Leuven, where he is heads a research team working in the area of numerical algorithms and signal processing for digital communications, wireless communications, DSL, and audio signal processing.

Dr. Moonen received the 1994 KU Leuven Research Council Award, the 1997 Alcatel Bell (Belgium) Award (with P. Vandaele), the 2004 Alcatel Bell (Belgium) Award (with R. Cendrillon), and was a 1997 "Laureate of the Belgium Royal Academy of Science." He received a journal Best Paper award from the IEEE TRANSACTIONS ON SIGNAL PROCESSING (with G. Leus) and from Elsevier *Signal Processing* (with S. Doclo). He was chairman of the IEEE Benelux Signal Processing Chapter (1998–2002), and is currently Past-President of European Association for Signal Processing (EURASIP) and a member of the IEEE Signal Processing Society Technical Committee on Signal Processing for Communications. He served as Editor-in Chief for the EURASIP *Journal on Applied Signal Processing* (2003–2005), and has been a member of the editorial board of *Integration*, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II (2002–2003) and IEEE SIGNAL PROCESSING MAGAZINE (2003–2005) and *Integration, the VLSI Journal*. He is currently a member of the editorial board of EURASIP *Journal on Advances in Signal Processing*, EURASIP *Journal on Wireless Communications and Networking*, and *Signal Processing*.